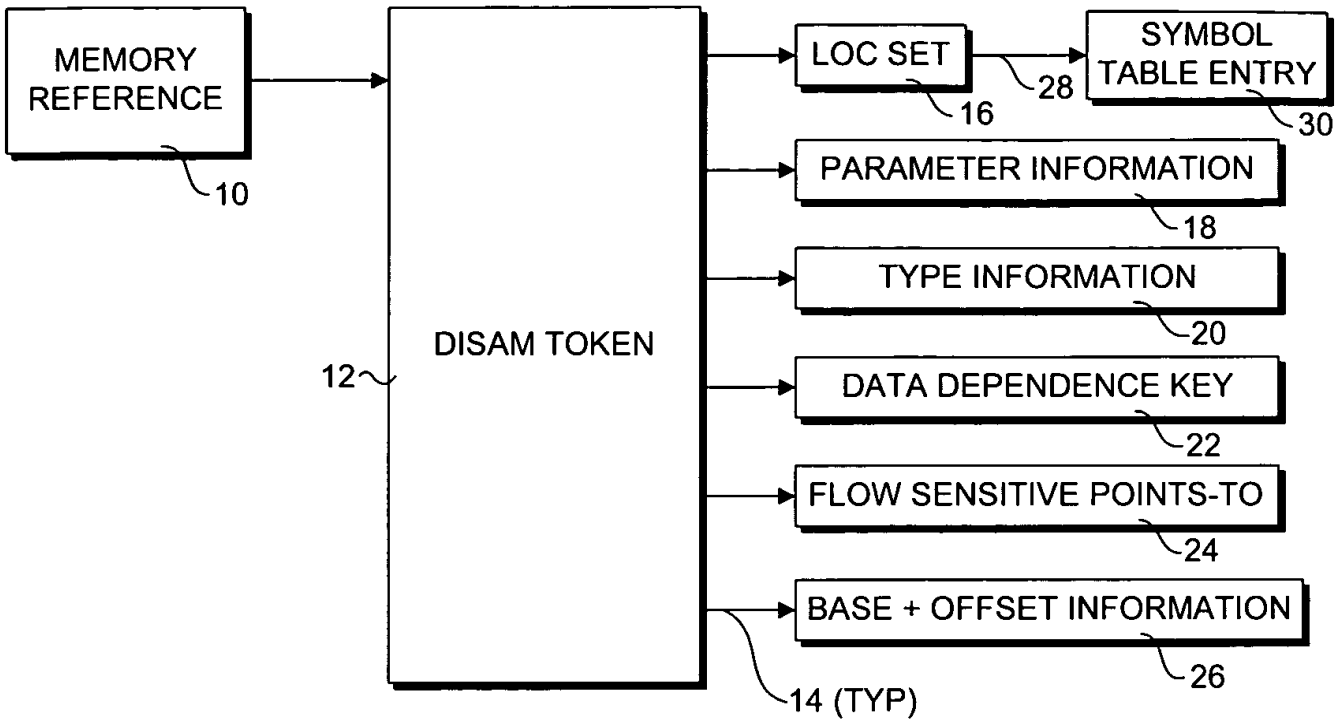

MEMORY REFERENCE
10
DISAM TOKEN
12
LOC SET
16
28
SYMBOL TABLE ENTRY
30
PARAMETER INFORMATION
18
TYPE INFORMATION
20
DATA DEPENDENCE KEY
22
FLOW SENSITIVE POINTS-TO
24
BASE + OFFSET INFORMATION
26
14 (TYP)

*FIG. 1*

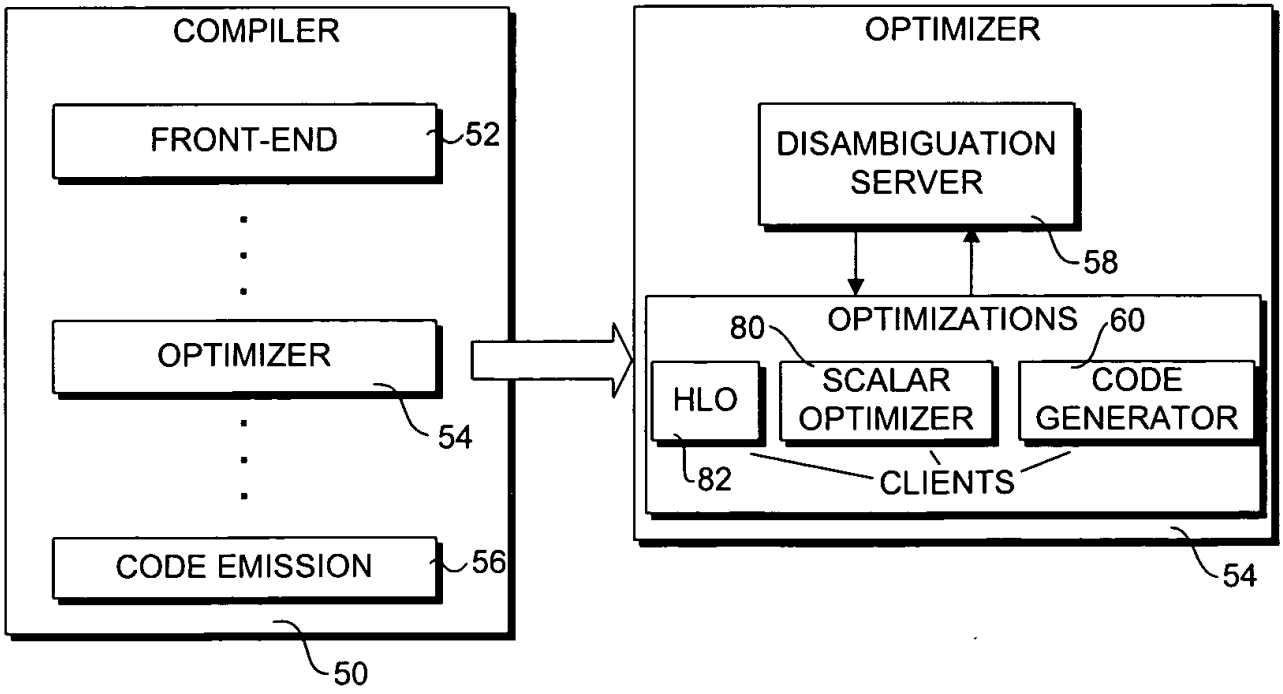

COMPILER
FRONT-END
52
OPTIMIZER
54
CODE EMISSION
56
50
OPTIMIZER
DISAMBIGUATION SERVER
58
OPTIMIZATIONS
60
80
HLO
SCALAR OPTIMIZER
CODE GENERATOR
82
CLIENTS
54

*FIG. 5*

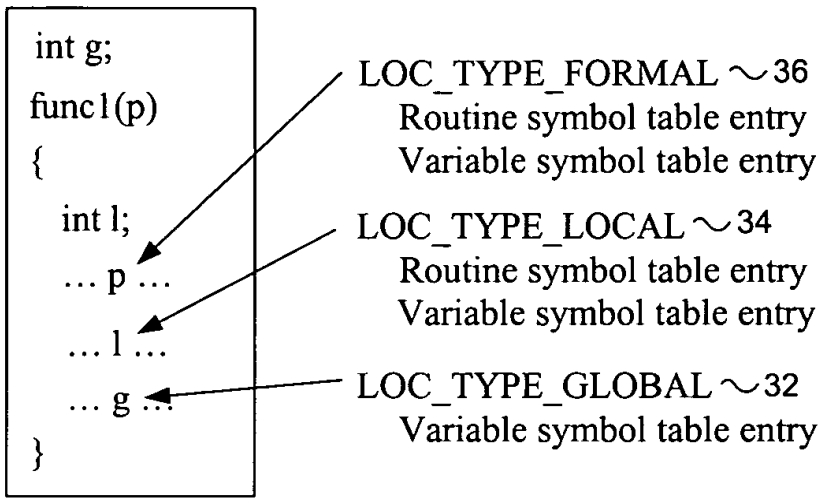



*FIG. 2A*

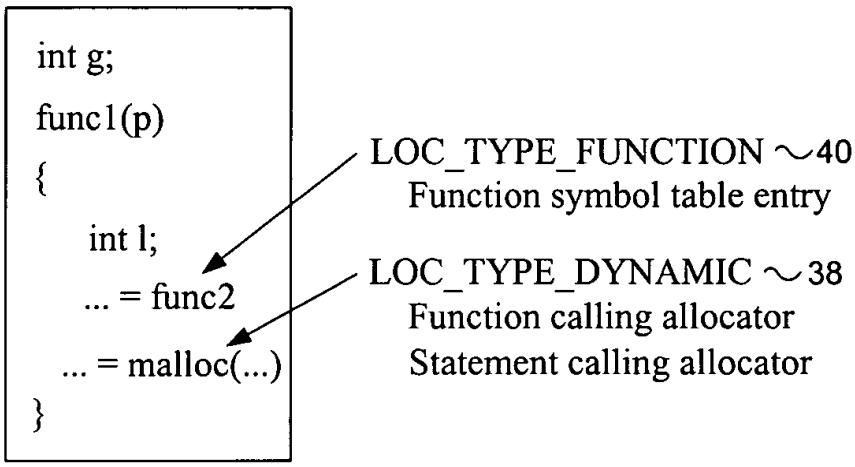



*FIG. 2B*

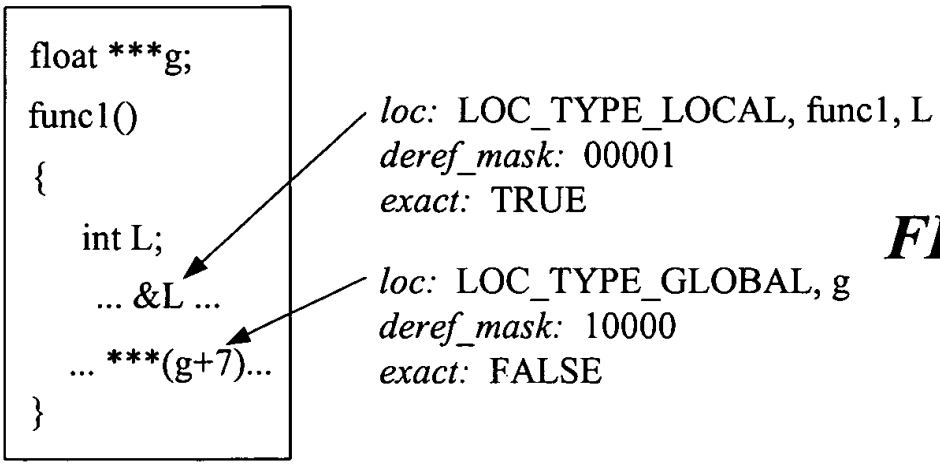
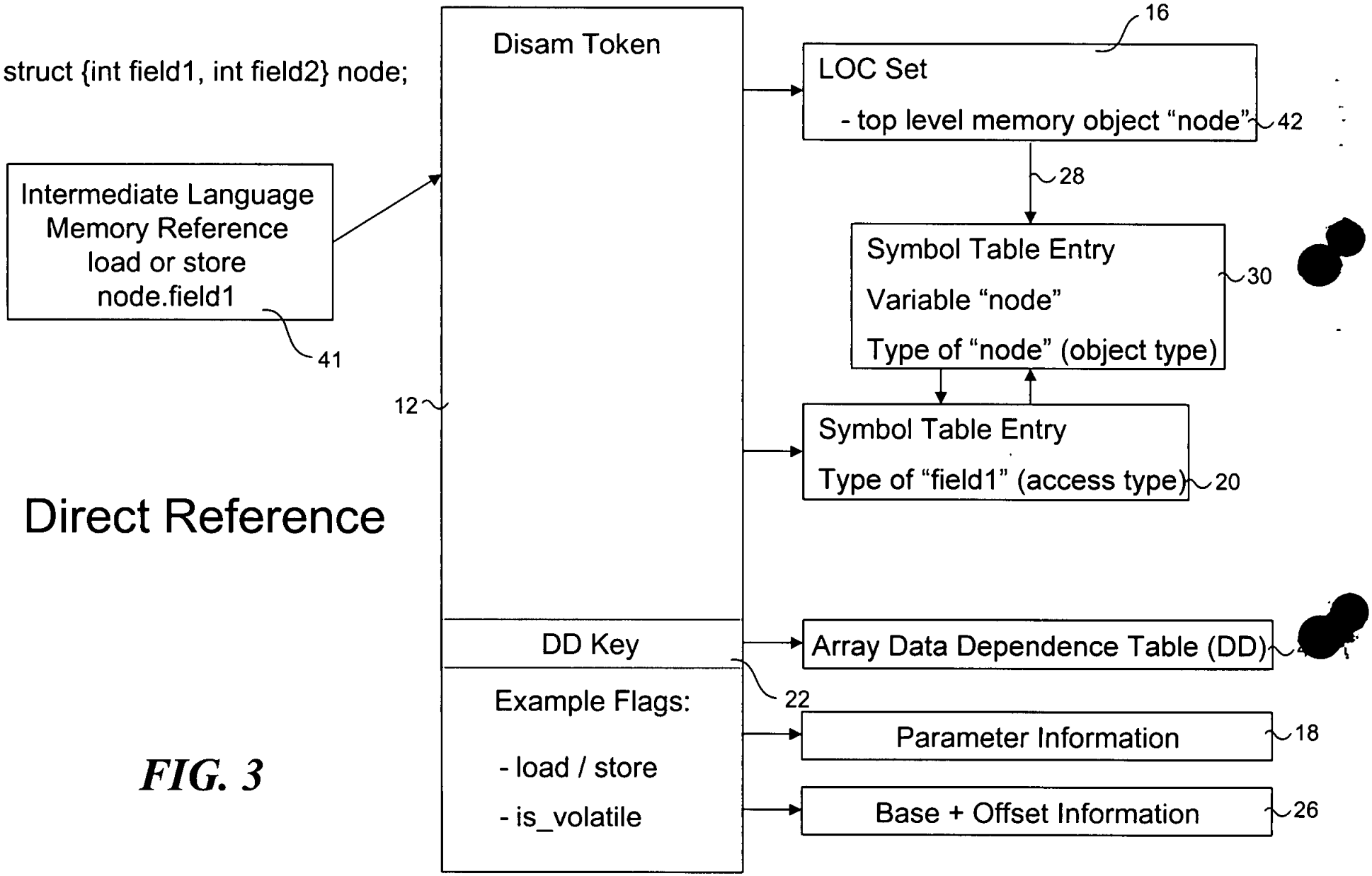



*FIG. 2C*




Direct Reference

*FIG. 3*

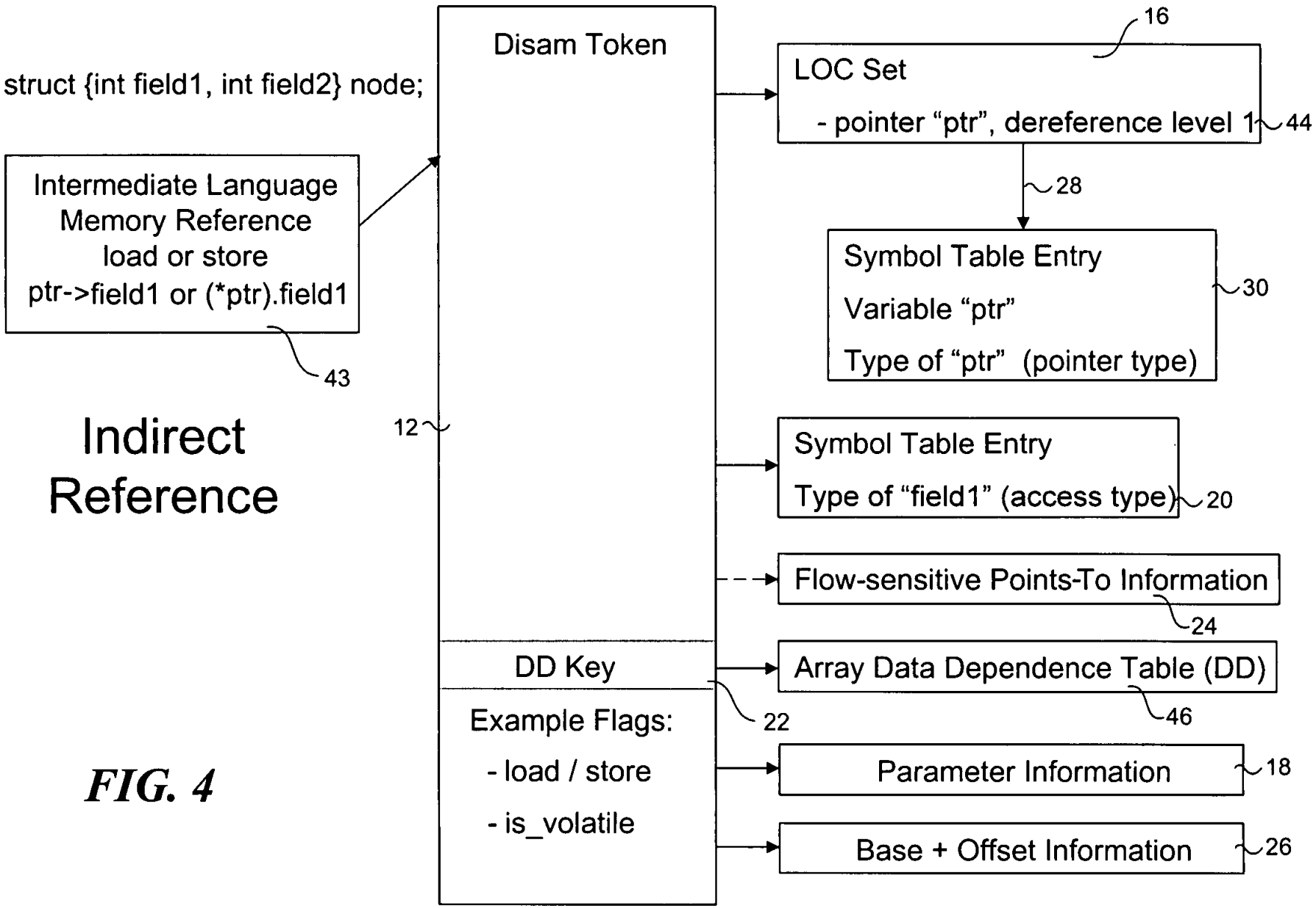

struct {int field1, int field2} node;
Intermediate Language
Memory Reference
load or store
ptr->field1 or (*ptr).field1
43
Indirect
Reference
12
Disam Token
DD Key
Example Flags:
- load / store
- is_volatile
22
16
LOC Set
- pointer "ptr", dereference level 1
44
28
Symbol Table Entry
Variable "ptr"
Type of "ptr" (pointer type)
30
Symbol Table Entry
Type of "field1" (access type)
20
Flow-sensitive Points-To Information
24
Array Data Dependence Table (DD)
46
Parameter Information
18
Base + Offset Information
26

*FIG. 4*

Toggled afterbed

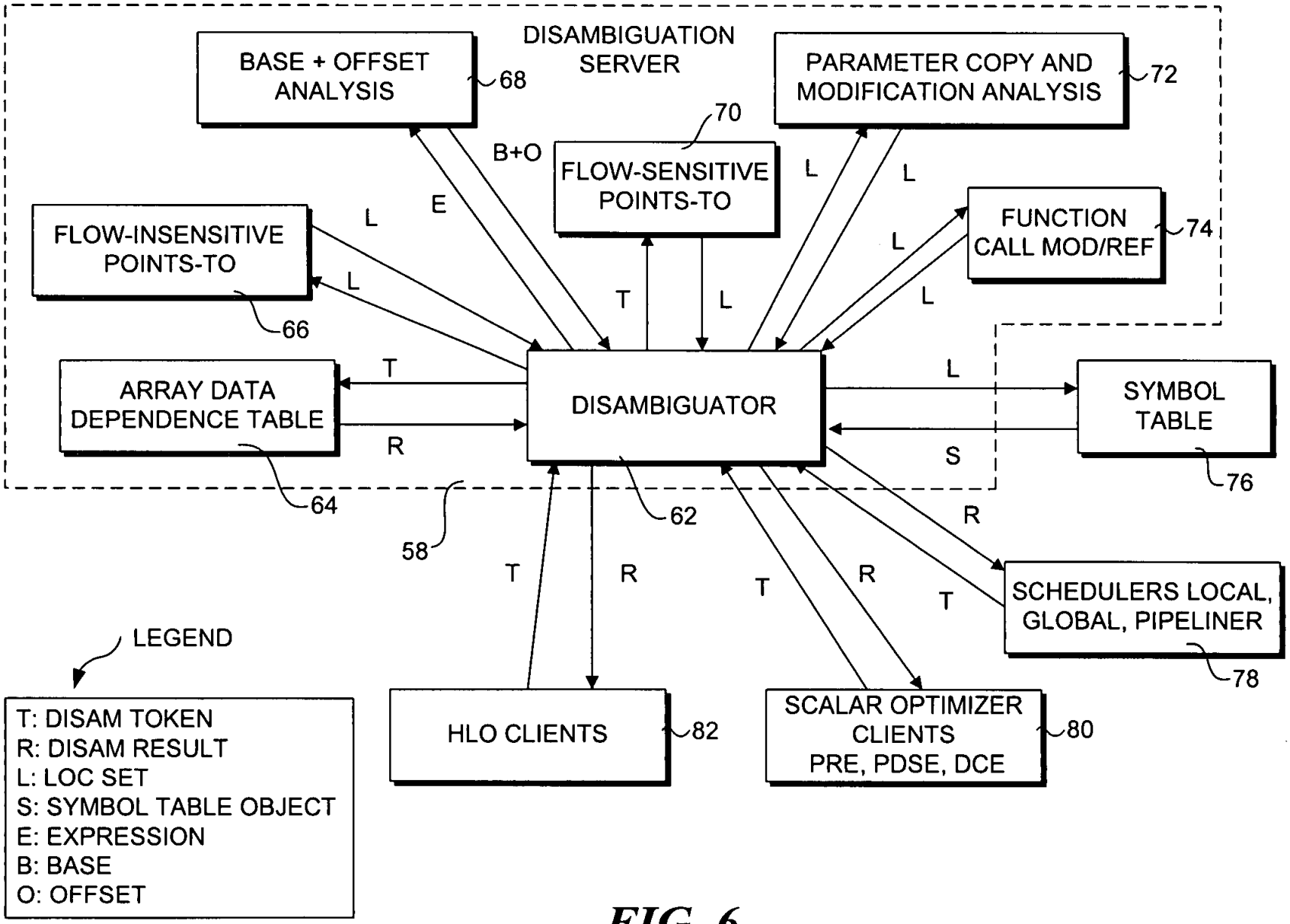



*FIG. 6*

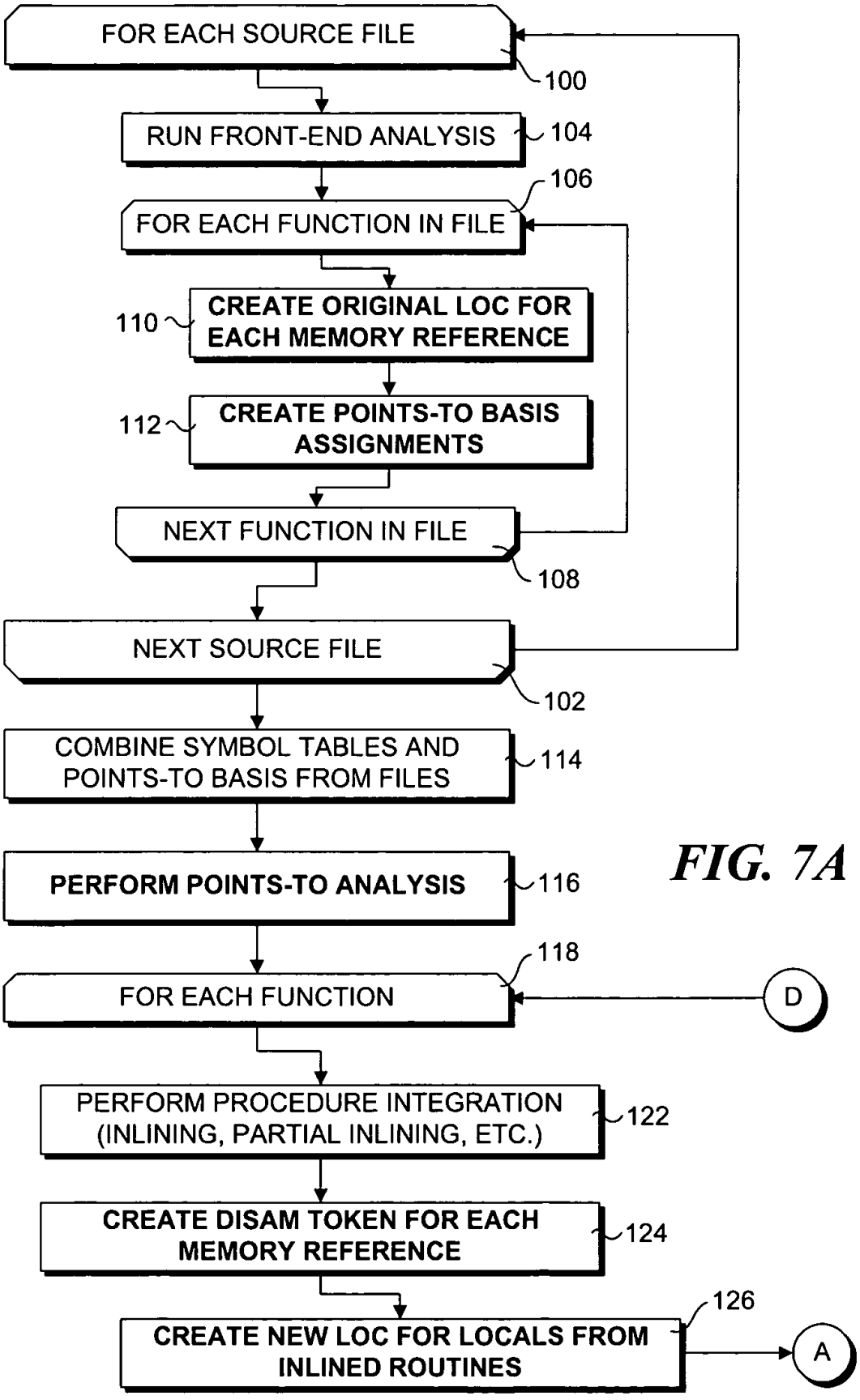

FOR EACH SOURCE FILE
100
RUN FRONT-END ANALYSIS
104
106
FOR EACH FUNCTION IN FILE
110
CREATE ORIGINAL LOC FOR EACH MEMORY REFERENCE
112
CREATE POINTS-TO BASIS ASSIGNMENTS
NEXT FUNCTION IN FILE
108
NEXT SOURCE FILE
102
COMBINE SYMBOL TABLES AND POINTS-TO BASIS FROM FILES
114
PERFORM POINTS-TO ANALYSIS
116
118
FOR EACH FUNCTION
D
PERFORM PROCEDURE INTEGRATION (INLINING, PARTIAL INLINING, ETC.)
122
CREATE DISAM TOKEN FOR EACH MEMORY REFERENCE
124
126
CREATE NEW LOC FOR LOCALS FROM INLINED ROUTINES
A

*FIG. 7A*

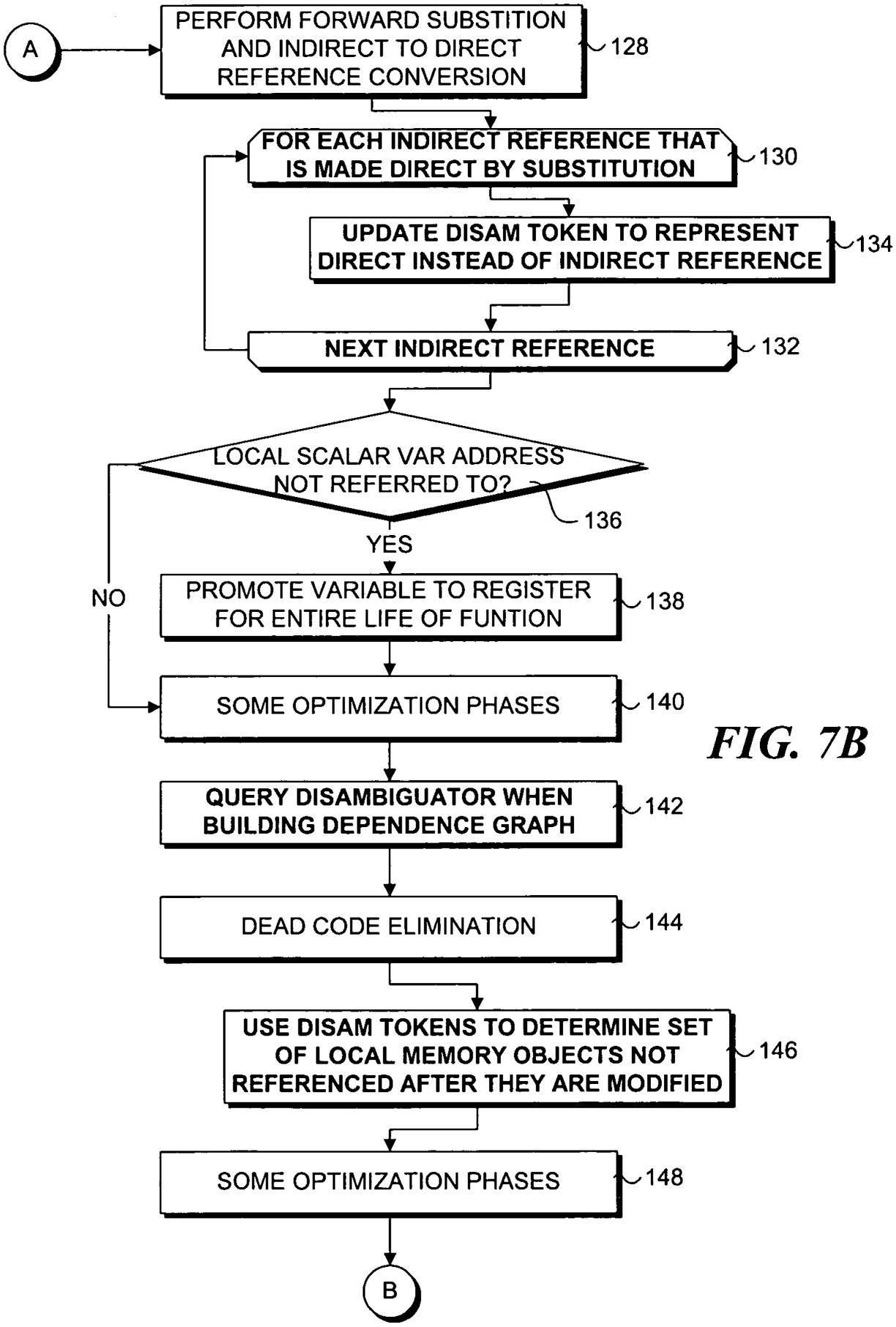

A
PERFORM FORWARD SUBSTITION AND INDIRECT TO DIRECT REFERENCE CONVERSION
128
FOR EACH INDIRECT REFERENCE THAT IS MADE DIRECT BY SUBSTITUTION
130
UPDATE DISAM TOKEN TO REPRESENT DIRECT INSTEAD OF INDIRECT REFERENCE
134
NEXT INDIRECT REFERENCE
132
LOCAL SCALAR VAR ADDRESS NOT REFERRED TO?
136
YES
NO
PROMOTE VARIABLE TO REGISTER FOR ENTIRE LIFE OF FUNTION
138
SOME OPTIMIZATION PHASES
140
QUERY DISAMBIGUATOR WHEN BUILDING DEPENDENCE GRAPH
142
DEAD CODE ELIMINATION
144
USE DISAM TOKENS TO DETERMINE SET OF LOCAL MEMORY OBJECTS NOT REFERENCED AFTER THEY ARE MODIFIED
146
SOME OPTIMIZATION PHASES
148
B

*FIG. 7B*

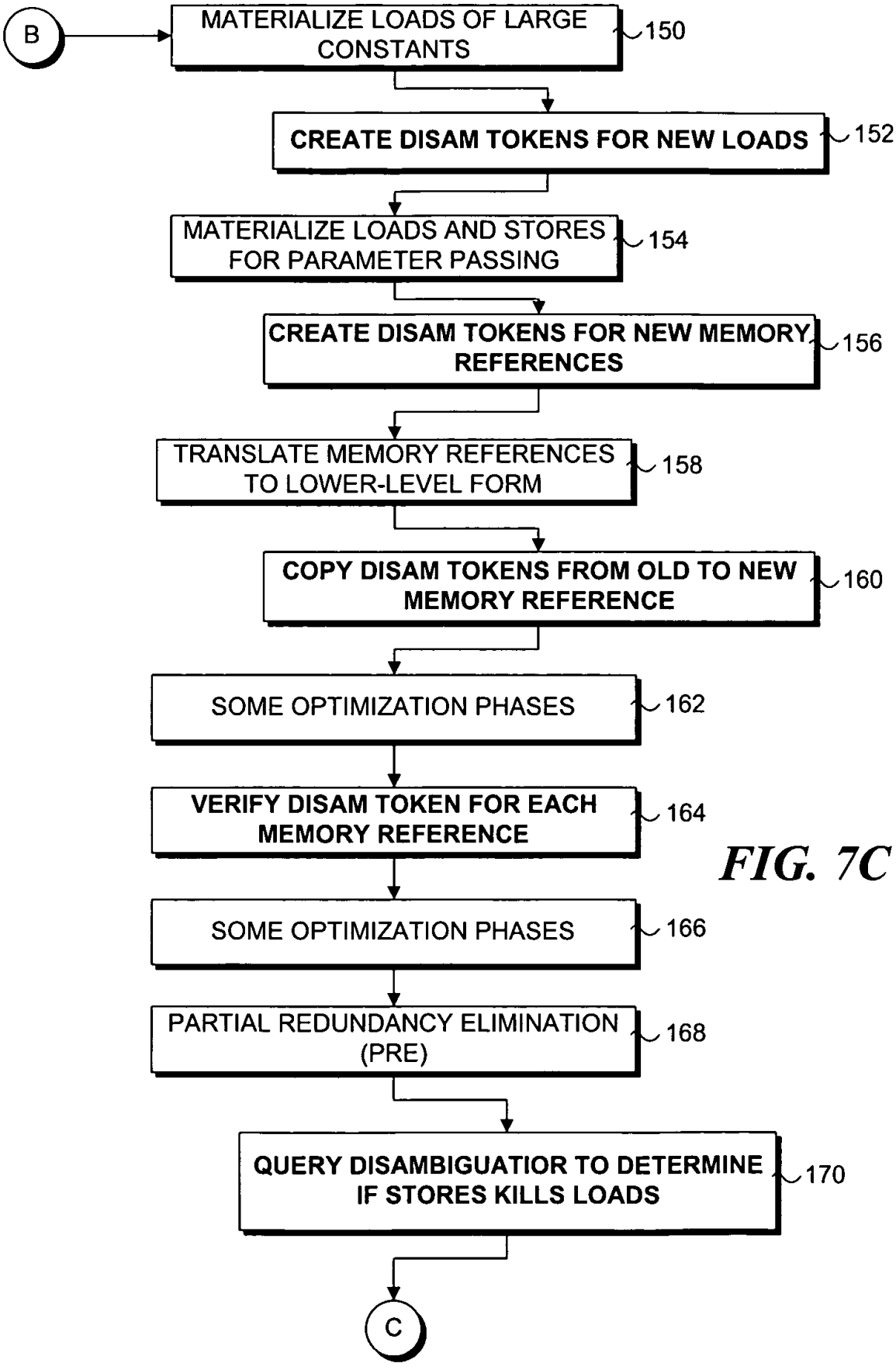

B
MATERIALIZE LOADS OF LARGE CONSTANTS
150
CREATE DISAM TOKENS FOR NEW LOADS
152
MATERIALIZE LOADS AND STORES FOR PARAMETER PASSING
154
CREATE DISAM TOKENS FOR NEW MEMORY REFERENCES
156
TRANSLATE MEMORY REFERENCES TO LOWER-LEVEL FORM
158
COPY DISAM TOKENS FROM OLD TO NEW MEMORY REFERENCE
160
SOME OPTIMIZATION PHASES
162
VERIFY DISAM TOKEN FOR EACH MEMORY REFERENCE
164
SOME OPTIMIZATION PHASES
166
PARTIAL REDUNDANCY ELIMINATION (PRE)
168
QUERY DISAMBIGUATIOR TO DETERMINE IF STORES KILLS LOADS
170
C

*FIG. 7C*

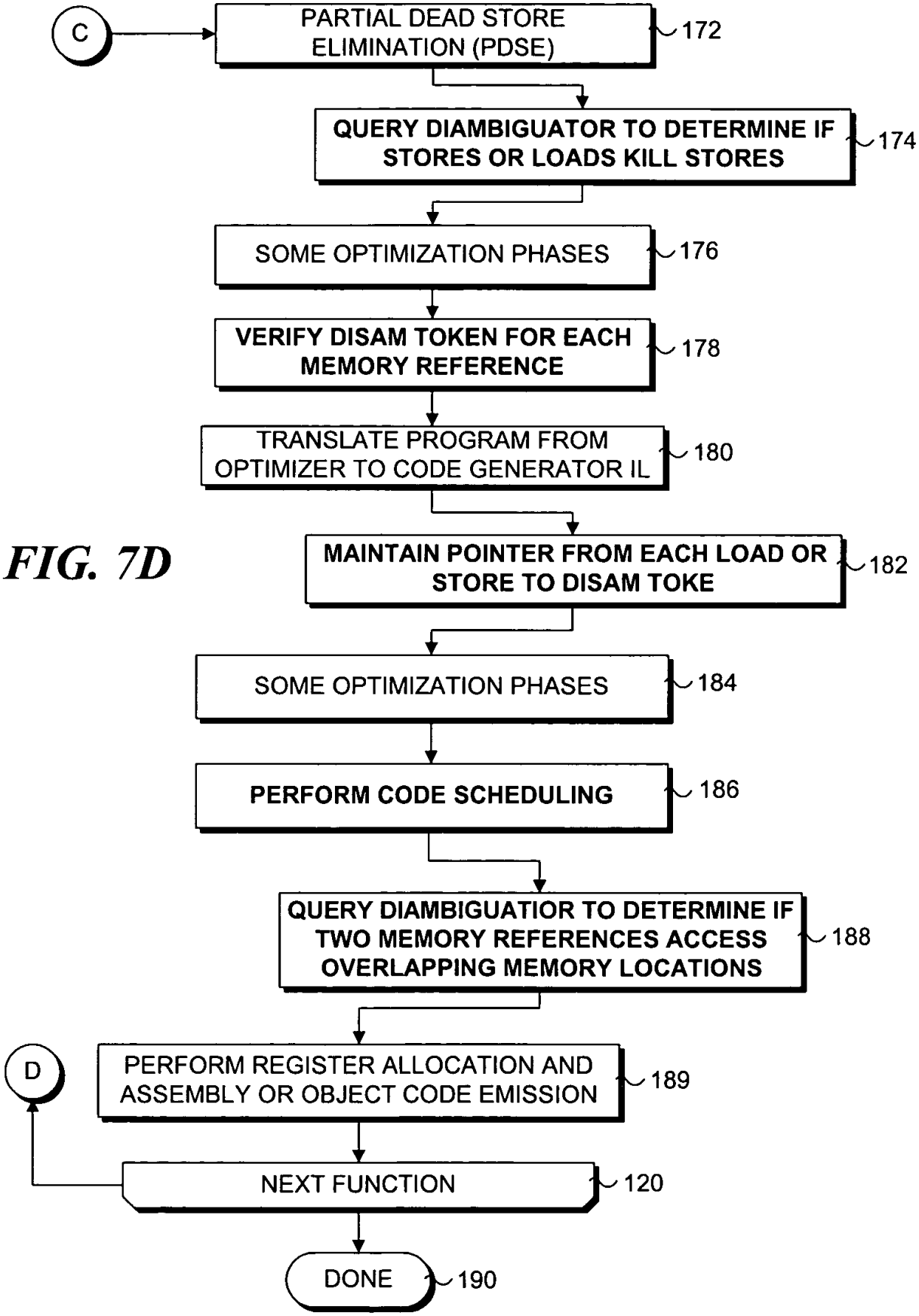

C
PARTIAL DEAD STORE ELIMINATION (PDSE)
172
QUERY DIAMBIGUATOR TO DETERMINE IF STORES OR LOADS KILL STORES
174
SOME OPTIMIZATION PHASES
176
VERIFY DISAM TOKEN FOR EACH MEMORY REFERENCE
178
TRANSLATE PROGRAM FROM OPTIMIZER TO CODE GENERATOR IL
180
MAINTAIN POINTER FROM EACH LOAD OR STORE TO DISAM TOKE
182
SOME OPTIMIZATION PHASES
184
PERFORM CODE SCHEDULING
186
QUERY DIAMBIGUATIOR TO DETERMINE IF TWO MEMORY REFERENCES ACCESS OVERLAPPING MEMORY LOCATIONS
188
D
PERFORM REGISTER ALLOCATION AND ASSEMBLY OR OBJECT CODE EMISSION
189
NEXT FUNCTION
120
DONE
190

*FIG. 7D*

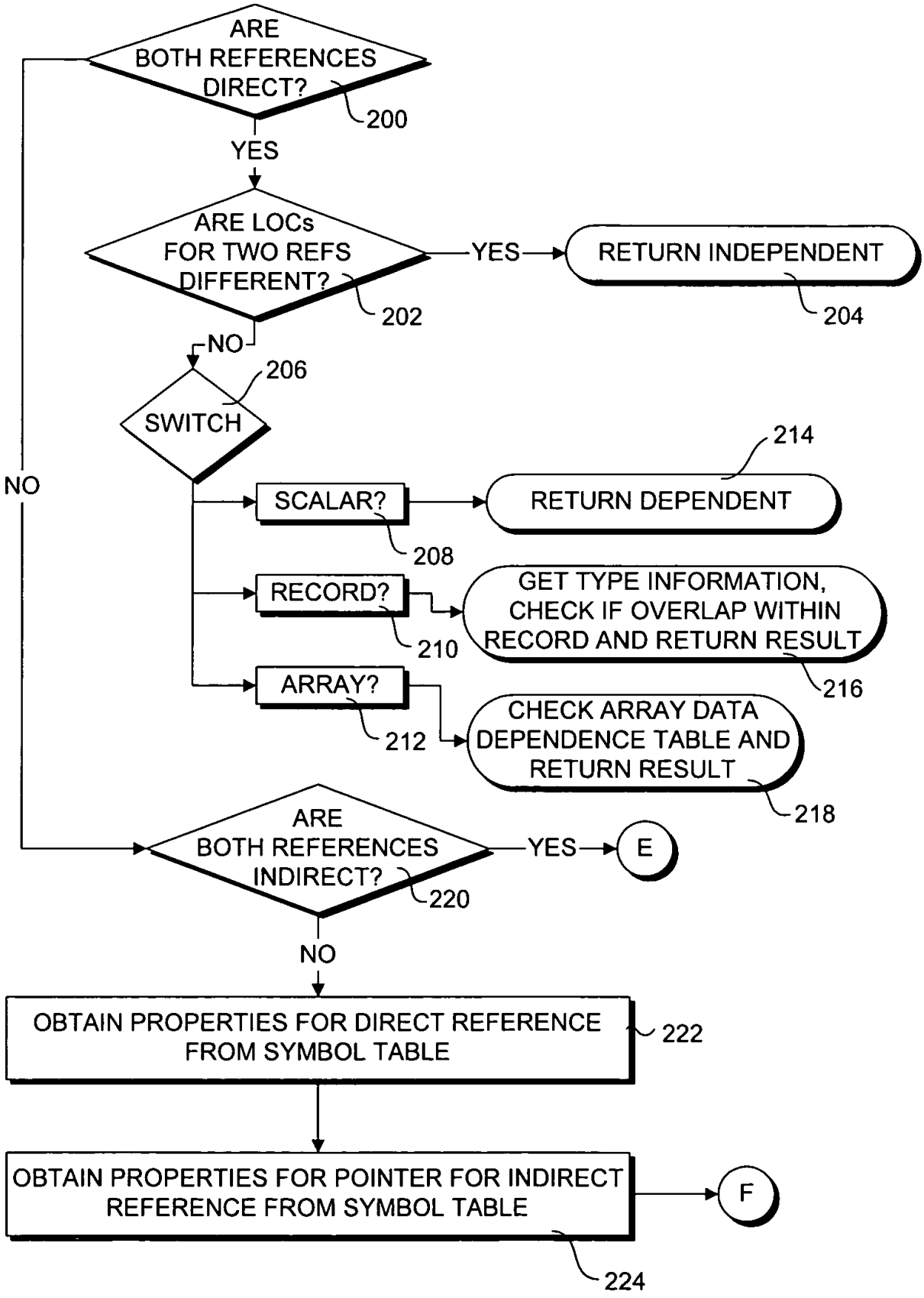

ARE BOTH REFERENCES DIRECT?
200
YES
ARE LOCs FOR TWO REFS DIFFERENT?
202
YES
RETURN INDEPENDENT
204
NO
206
SWITCH
NO
214
SCALAR?
208
RETURN DEPENDENT
RECORD?
210
GET TYPE INFORMATION, CHECK IF OVERLAP WITHIN RECORD AND RETURN RESULT
216
ARRAY?
212
CHECK ARRAY DATA DEPENDENCE TABLE AND RETURN RESULT
218
ARE BOTH REFERENCES INDIRECT?
220
YES
E
NO
OBTAIN PROPERTIES FOR DIRECT REFERENCE FROM SYMBOL TABLE
222
OBTAIN PROPERTIES FOR POINTER FOR INDIRECT REFERENCE FROM SYMBOL TABLE
F
224

*FIG. 8A*

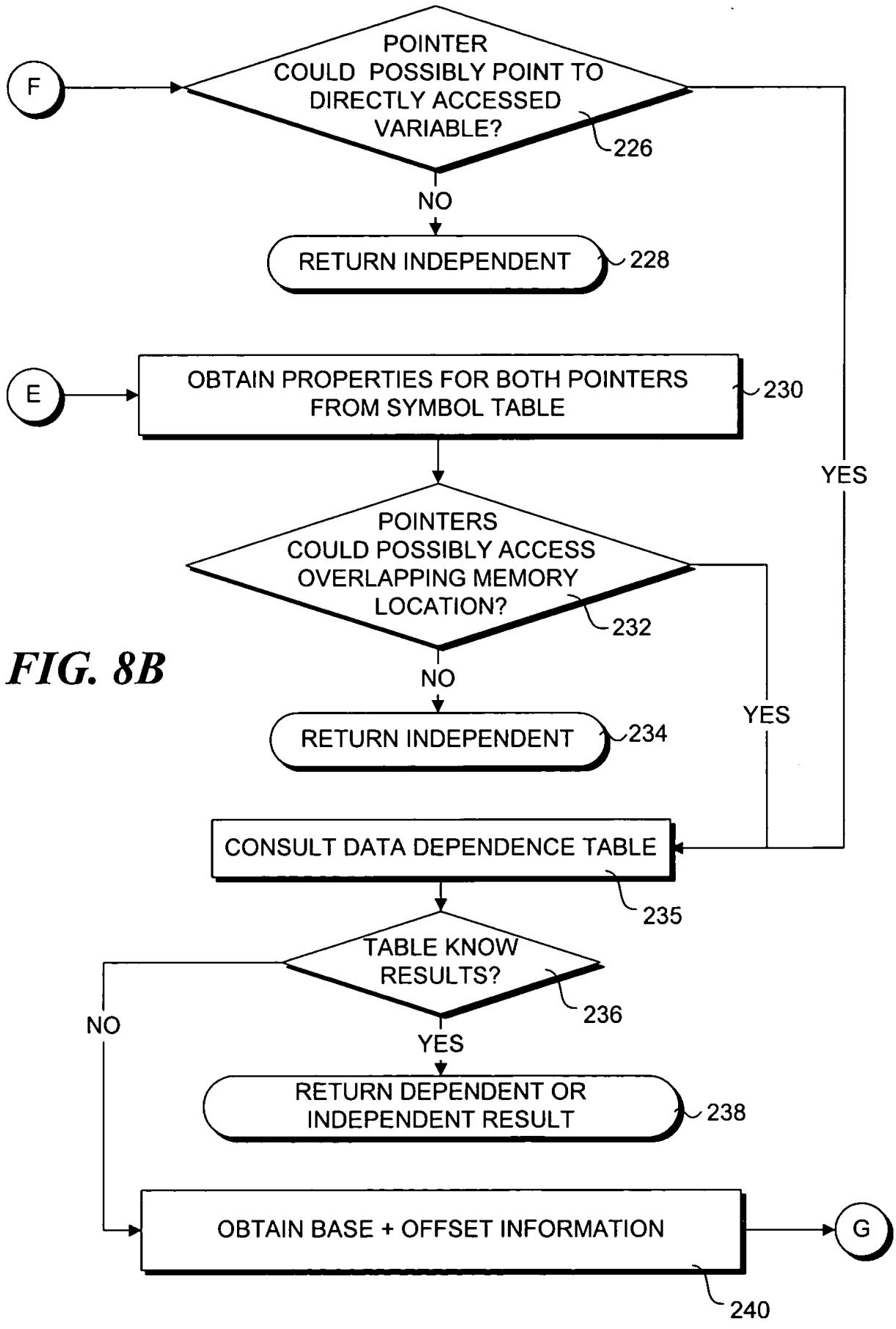

F
POINTER COULD POSSIBLY POINT TO DIRECTLY ACCESSED VARIABLE?
226
NO
RETURN INDEPENDENT
228
YES
E
OBTAIN PROPERTIES FOR BOTH POINTERS FROM SYMBOL TABLE
230
POINTERS COULD POSSIBLY ACCESS OVERLAPPING MEMORY LOCATION?
232
NO
YES
RETURN INDEPENDENT
234
CONSULT DATA DEPENDENCE TABLE
235
TABLE KNOW RESULTS?
236
NO
YES
RETURN DEPENDENT OR INDEPENDENT RESULT
238
OBTAIN BASE + OFFSET INFORMATION
240
G

*FIG. 8B*

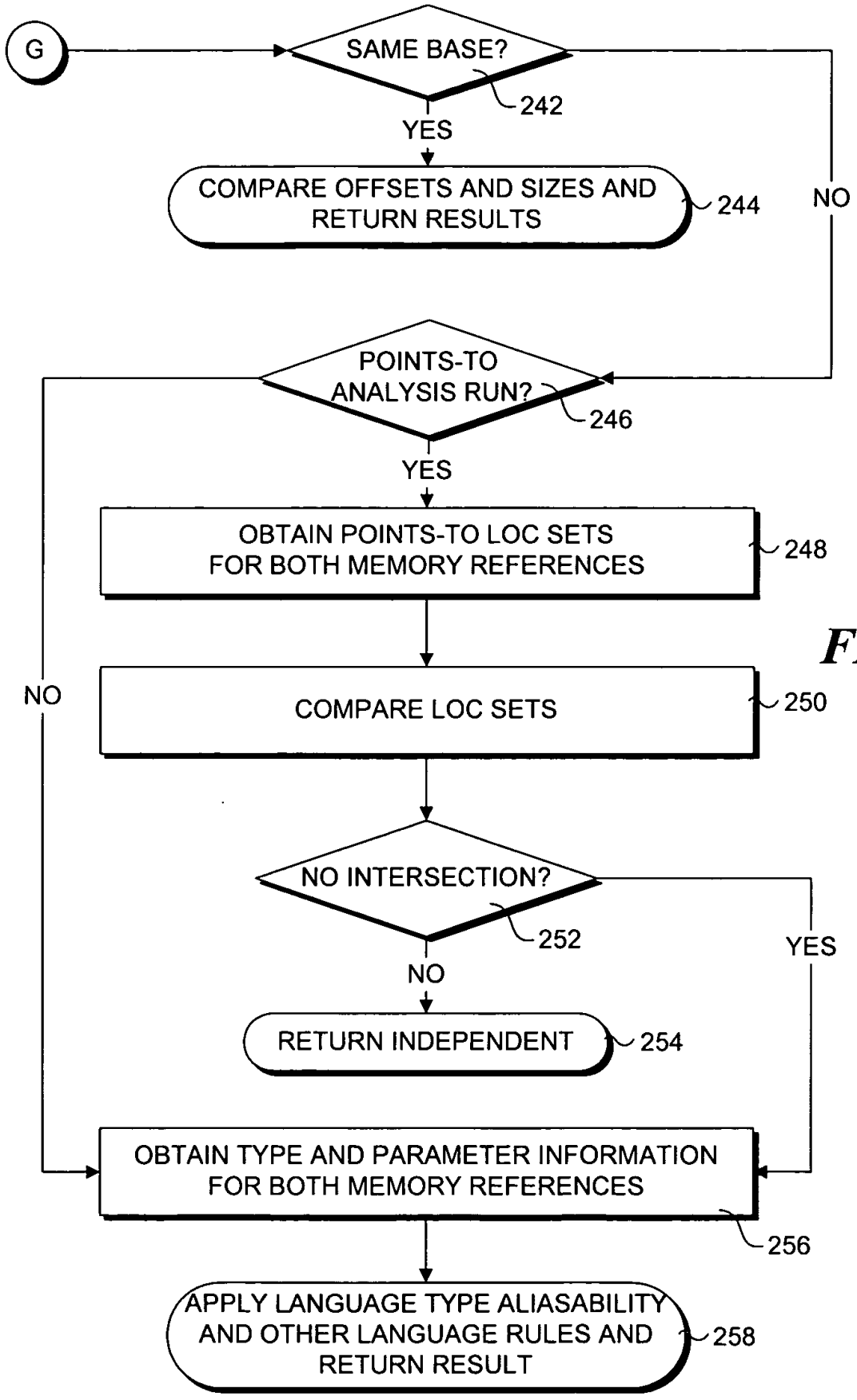

G
SAME BASE?
242
YES
NO
COMPARE OFFSETS AND SIZES AND RETURN RESULTS
244
POINTS-TO ANALYSIS RUN?
246
YES
NO
OBTAIN POINTS-TO LOC SETS FOR BOTH MEMORY REFERENCES
248
COMPARE LOC SETS
250
NO INTERSECTION?
252
YES
NO
RETURN INDEPENDENT
254
OBTAIN TYPE AND PARAMETER INFORMATION FOR BOTH MEMORY REFERENCES
256
APPLY LANGUAGE TYPE ALIASABILITY AND OTHER LANGUAGE RULES AND RETURN RESULT
258

*FIG. 8C*

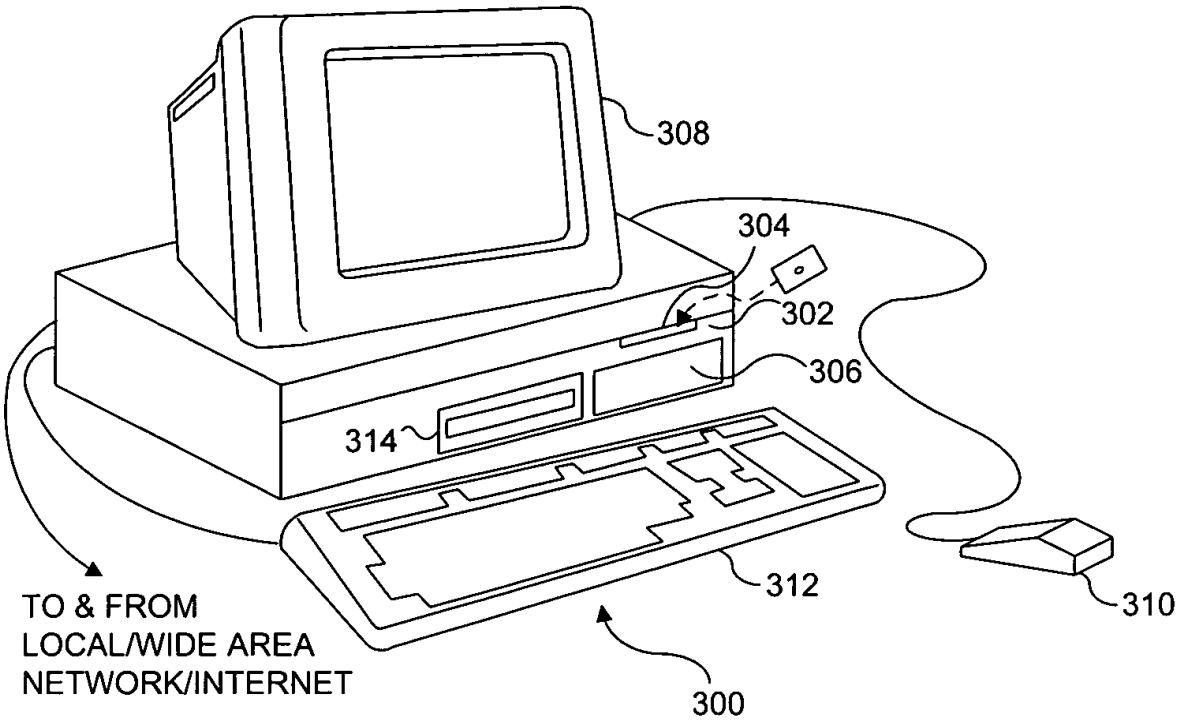

308
304
302
306
314
312
310
TO & FROM
LOCAL/WIDE AREA
NETWORK/INTERNET
300

*FIG. 9*